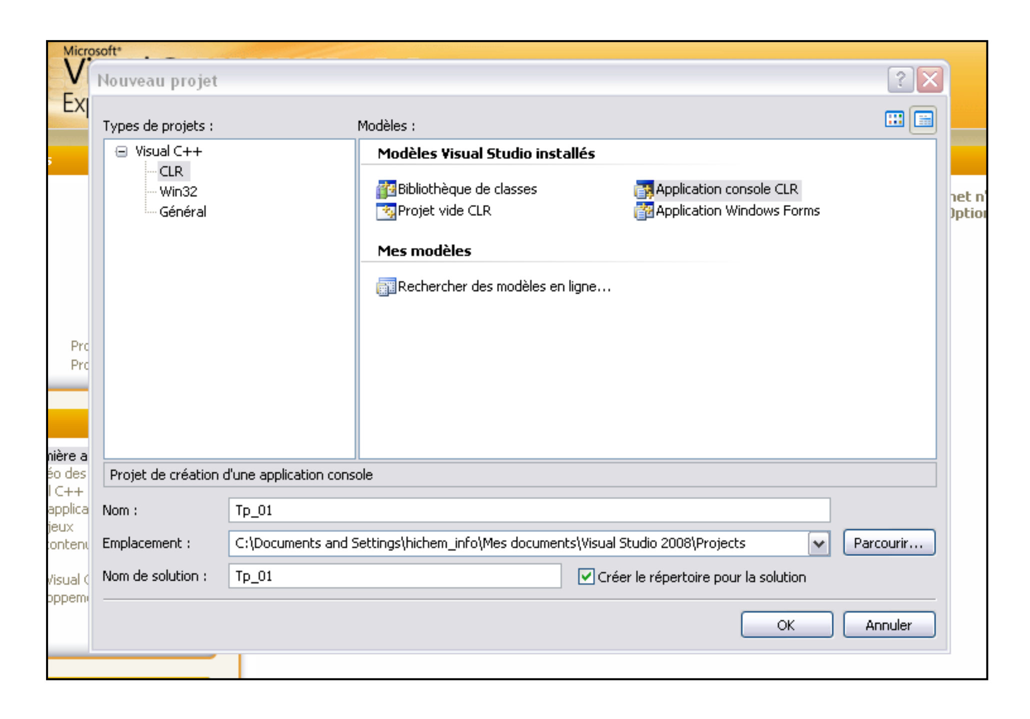
1- Table des matières



1- La classe string en C++ :

السلاسل في لغة C++ عبارة عن كلاس يحتوي علي الكثير من الدوال تسهل العمل مع السلاسل علي خلاف لغة C التي يعتبر التعامل مع السلاسل فيها صعبا إلى حد ما . توفر لغة C ++ مكتبة كاملة للتعامل مع السلاسل وهي :

```
#include <string>
```

تمكننا هده المكتب من القيام بعدة عمليات كالإسناد والحذف والتبديل وغيرها من العمليات التي سنعرفها لاحقا.

➢ La déclaration :

للتصريح بسلسلة عليك أن لا تنسى كتابة المكتبة التي ذكرنها سابقا. اعرف آن ذاكرتك قوية ....أنت (ة) أفضل مبرمج(ة) .

```
#include "stdafx.h"
#include <iostream>
#include <string>

using namespace System;
using namespace std;

int main(array<System::String ^> ^args)
{
	string str;
	cin >> str;
	cout<<" my String : "<<str<<endl;
	Console::ReadKey();
    return 0;
}
```

في هدا المثال قمنا بالتصريح بسلسلة ثم قرانها بواسطة دالة القراءة في السي بلس بلس وبعد دلك إظهار السلسلة. أتمنى أن يكون هدا العمل قد أعجبكم

➢ initialisation et affectation d'une string :

```
#include "stdafx.h"
#include <iostream>
#include <string>

using namespace System;
using namespace std;

int main(array<System::String ^> ^args)
{
	string str1="hichem";
	string str2="jijel";
	cout<<" my String1 : <<str1<<endl;
	cout<<" my String2 : "<<str2<<endl;
	Console::ReadKey();
    return 0;
}
```

```
my String1 : hichem

my String1 : jijel
```

كان هدا التصريح بسلسلتين مع إعطاء كل واحدة منهما سلسلة ابتدائية ثم إظهار السلسلتين كما في الصورة .

لا تنسى حلمك في أن تصبح المبرمج الأول .....

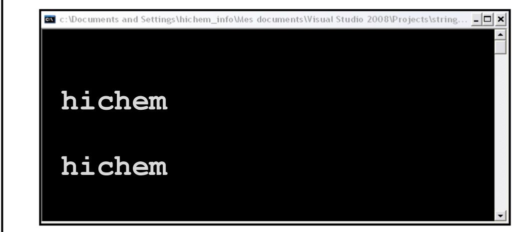
```
// stringg.cpp : fichier projet principal.

#include "stdafx.h"
#include <iostream>
#include <string>

using namespace System;
using namespace std;

int main(array<System::String ^> ^args)
{
      string str1="hichem";
      string str2;
      //---------------------
      str2 = str1;
      //---------------------
      cout<<" my String1 : "<<str1<<endl;
      cout<<" my String2 : "<<str2<<endl;
      Console::ReadKey();
    return 0;
}
```

ستشاهد نفس الصورة السابقة ولكن نفس السلسلة الأولى تظهر مرتين وهذا لأننا قمنا بإسناد السلسلة الأولى إلي السلسلة الثانية وبالتالي صار لكل منهما نفس الحروف .



➢ Concaténation des string :

```
#include "stdafx.h"
#include <iostream>
#include <string>

using namespace System;
using namespace std;
int main(array<System::String ^> ^args)
{
      string str1="c++";
      string str2="hichem";
      string s;
      //---------------------
      s = str1 + str2;
      //---------------------
      cout<<" my String : "<<s<<endl;
      Console::ReadKey();
    return 0;
}
```

عملية دمج السلاسل في السي بلس بلس سهلة جدا كون السلسلة عبارة عن كلاس به مجموعة من الدوال والمعاملات تسمح لنا بالتعامل مع السلاسل بكل مرونة.

أولا باستعمال المعامل +

نتيجة المثال

```
my String : c++hichem
```

```
#include "stdafx.h"
#include <iostream>
#include <string>

using namespace System;
using namespace std;
int main(array<System::String ^> ^args)
{
      string s;
      //---------------------
      s.append("c++hichem", 4);
      //---------------------
      cout<<" my String : "<<s<<endl;
      Console::ReadKey();
    return 0;
}
```

ثانيا باستعمال الدالة append(string , int)

هده الدالة يلزمها متغيرين الأول هو عبارة عن سلسلة والثاني عبارة عن عدد صحيح. تقوم هده الدالة بأخذ عدد من الحروف من السلسلة التي أرسلت إليها كمتغير بمقدار العدد الذي أرسل كمتغير ثاني ووضعها في السلسلة التي قامت باستدعاء هده الدالة.

نتيجة المثال

```
my String : c++h
```

➢ La langur de String :

```
#include "stdafx.h"
#include <iostream>
#include <string>

using namespace System;
using namespace std;

int main(array<System::String ^> ^args)
{
string s="merieme";
cout<<" * La langur de String : "<<s.length()
<<endl;
Console::ReadKey();
    return 0;
}
```

```
* La langur de String : 7
```

➢ Échange du contenu de deux string :

```
// stringg.cpp : fichier projet
principal.

#include "stdafx.h"
#include <iostream>
#include <string>

using namespace System;
using namespace std;

int main(array<System::String ^> ^args)
{
      string s1="hichem";
      string s2="merieme";
      //---------------------
      s1.swap(s2);
      //---------------------
      cout<<" my String1 : "<<s1<<endl;
      cout<<" my String2 : "<<s2<<endl;

      Console::ReadKey();
    return 0;
}
```

```
my String1 : merieme

my  String2 : hichem
```

في هدا المثال استعملنا دالة التبديل بين سلسلتين . التي تحتاج إلي متغير واحد وهو السلسلة التي نريد إجراء التبديل عليها .

```
Str1.swap(str2) ;
```

➢ Fonctions de recherche dans les string :

find	تسمح هده السلسلة بالبحث عن سلسلة داخل سلسلة وتعيد موضع أول حرف في السلسلة الداخلية
rfind	نفس عمل الدالة السابقة لكن بداية البحث تكون من أخر السلسلة التي نبحث داخلها وتعيد لنا موضع أخر حرف في السلسلة الداخلية .

```
// stringg.cpp : fichier projet principal.

#include "stdafx.h"
#include <iostream>
#include <string>

using namespace System;
using namespace std;

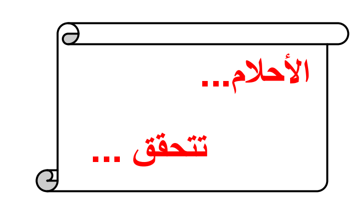
int main(array<System::String ^> ^args)
{
       int p;

       string s1="hichemC++Java";
       //---------------------
       p=s1.find("C++");
       //---------------------
       cout<<" index = "<<p<<endl;

       Console::ReadKey();
     return 0;
}
```



كما رأيت تم إظهار الرقم 6 لان 6 هو موضع أول حرف في السلسلة الداخلية لا تنسى أن بداية العد في السي بلس بلس تبدأ من الصفر .



➢ Insertion dans une string :

```
#include "stdafx.h"
#include <iostream>
#include <string>

using namespace System;
using namespace std;

int main(array<System::String ^> ^args)
{
       string s1="hichem";
       //---------------------
       s1.insert(2,"C++Java");
       //---------------------
       cout<<" my string : "<<s1<<endl;

       Console::ReadKey();
     return 0;
}
```

```
hiC++Javachem
```

تحتاج هده الدالة إلي متغيرين الأول هو موضع الإدخال والثاني هو السلسلة التي نريد إدخالها .

في المثال قمنا بادخال سلسلة إلي سلسلة أخري بداية من الموضع 2 فكانت النتيجة كما في الصورة السابقة .

لإضافة سلسلة إلى أخر سلسلة أخرى نستعمل الدالة

```
string s1= "hichemC++Java";
s1.append("C#");
cout<<" my string :"<<s1<<endl;

// hichemC++JavaC#
```

➢ Suppression de string :

Suppression de sous -string :

```
string s1= "hichemC++Java";
//---------------------
s1.erase(6,3);
//---------------------
cout<<" my string : "<<s1<<endl;
```

hichemJava

دالة مسح سلسلة داخل سلسلة تحتاج إلي متغيرين الأول هو موضع بدية المسح والثاني هو عدد الأحرف التي نريد مسحها .

مسح كل السلسلة `s1.clear();`

➢ Remplacement d'une sous-string dans une string :

```
int p;
string s1= "hichemC++Java";
p=s1.find("C++");
//---------------------
s1.replace(p,3,"C#");
//---------------------
cout<<" my string : "<<s1<<endl;
```

hichemC#Java

في المثال بحثنا عن موضع سلسلة داخلية ثم قمنا باستبدالها بسلسلة أخري.

دالة الاستبدال تحتاج إلي ثلاث متغيرات الأول بداية السلسلة التي نريد استبدالها والثاني طول هده السلسلة أما الثالث فهي السلسلة التي نريد إظهارها .

➢ Copie une sous-string dans une string :

```
string s1,s= "hichemC++Java";
//---------------------
s1=s.substr(1,7);
//---------------------
cout<<" my string :"<<s1<<endl;
```

ichemC+

لنسخ جزء من سلسلة إلي سلسلة أخري نستعمل الدالة كما في المثال وهي تستقبل متغيرين الأول موضع بداية النسخ والثاني موضع نهاية النسخ . أو مباشرة مثل

```
string s= "hichemC++Java";
string s1(s,1,7); //السلسلة كل string s1(s);
cout<<" my string : "<<s1<<endl;

// string s1(s.begin(),s.end()); نسخ السلسلة من البداية حتي النهاية
```

➢ Autre opérations sur les strings:

`int t = str.size();`	معرفة حجم السلسلة في الذاكرة بال_ Octect .
`char c = str[i];`	الوصول إلي حرف في سلسلة.
`int cp = str.capacity();`	سعة سلسلة .
`int n = str.max_size();`	معرفة الحجم الأقصى للسلسلة قبل إعادة حجز الذاكرة من جديد .
`getline(cin,str,'\n');`	إدخال سلسلة تحتوي علي فراغات .

2. Les fichiers :

التعامل مع الملفات في السي بلس بلس ليس صعبا كما يتخيل البعض . فهناك الكثير من الأشياء توفرها لغة السي بلس بلس تجعل من التعامل مع الملفات أمرا سهلا وبسيطا جدا . سنتعرف في هدا الدرس عن كيفية فتح وقراءة ملف نصي .

➢ L'en-tête fstream: #include <fstream>

```
// stringg.cpp : fichier projet principal.

#include "stdafx.h"
#include <iostream>
#include <string>
#include <fstream>

using namespace System;
using namespace std;
```

هده المكتبة يجب تضمينها قبل بدا العمل مع الملفات لأنها هي التي تحتوي علي دوال الإدخال والإخراج التي نحتاجها حتى نتمكن من الكتابة علي ملف نصي أو قراءة ملف نصي .

➢ Écrire dans un fichier:

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <fstream>

using namespace System;
using namespace std;

int main(array<System::String ^> ^args)
{
      ofstream monFlux; //التصريح بالملف
      string s="jijel";
      int n;
      n=145;
    monFlux.open("D:/fi_c++.txt",ios::app|ios::out);
   //تحديد نوع الملف مع تحديد المسار الدي يتواجد فيه الملف
     if(monFlux) //للتأكد من ان الملف قد فتح
       {
//الكتابة داخل الملف
         monFlux <<"hichem"<<endl; //كتابة سلسلة نصية
          monFlux <<"merieme"<<endl;
           monFlux << s <<endl; //كتابة سلسلة نصية مصرح بها مسبقا
              monFlux << n <<endl; //كتابة عدد صحيح مصرح به مسبقا
               monFlux <<12478<<endl; //كتابة عدد صحيح
    monFlux.close();//غلق الملف بعد الانتهاء من الكتابة
    cout << "ok:"<<endl; //للاشارة فقط انا انتهينامن الكتابة
            }
           else
          {
//في حالة عدم فتح الملف تخبرنا هده الرسالة
           cout << "ERREUR: Impossible d'ouvrir le fichier." << endl;
        }

 Console::ReadKey();
    return 0;
}
```

fi_c++ - Bloc-notes
Fichier Edition Format Affichage ?

```
hichem
merieme
jijel
145
12478
```

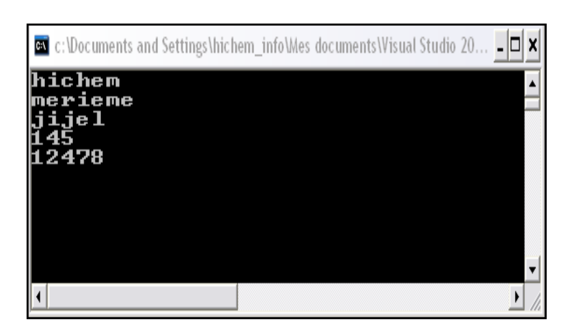
ios ::app	تكون عملية الإدخال إلي الملف من النهاية .
ios ::ate	يقوم بالقراءة أو الكتابة من نهاية الملف .
ios::trunce	في حالة وجود الملف يقوم بحذف محتوياته .
ios::noceate	اذا كان الملف غير موجود تفشل عملية الفتح .
ios ::noreplace	اذا كان الملف موجود تفشل عملية الفتح .
ios ::in	فتح الملف للقراءة .
ios ::out	فتح الملف للكتابة .
ios::binary	يستعمل لفتح الملفات الثنائية .

➢ Lecture d'un fichier:

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <fstream>

using namespace System;
using namespace std;

int main(array<System::String ^> ^args)
{
      string s;
      fstream monFlux("D:/fi_c++.txt",ios::in); //التصريح بملف للقراءة
      if(monFlux) //التأكد من فتح الملف
      {
       monFlux.seekg(0); //وضع مؤشر الملف في البداية
       while(!monFlux.eof())//حلقة تكرارية حتى نقرىء كل الملف حتى النهاية
        {
            getline(monFlux, s); //قراءة سطر من الملف ووضعه في سلسلة نصية
            //monFlux >>s; //قراءة سطر من الملف ووضعه في سلسلة نصية
          cout<< s<<endl; //اظهار السطر الدي قرأنها على الشاشة
         }
       monFlux.close();//غلق الملف بعد نهاية القراءة
      }
   else//في حالة عدم فتح الملف تخبرنا هده الرسالة
     {
      cout << "ERREUR: Impossible d'ouvrir le fichier en lecture." <<endl;
     }
 Console::ReadKey();
    return 0;
}
```



➢ Exemple sur les fichiers :

```
#include "stdafx.h"
#include <iostream>
#include <fstream>
#include <string>

using namespace System;
using namespace std;
//---------------------------------------------------------------------------
struct per
{
int _n;
string _nom,_prenom;
};
//---------------------------------------------------------------------------
void home ()
{

cout<<"\n";
cout<<"                  +-----------------------------------------+\n";
cout<<"                  | 1.Pour saisie les perssonts          : |\n";
cout<<"                  | 2.Pour affiche tous les perssonts    : |\n";
cout<<"                  | 3.Pour exite                         : |\n";
cout<<"                  +-----------------------------------------+\n";
}
//---------------------------------------------------------------------------
void ec_f(per _ele)
{
   ofstream monFlux;
        string s="-------------------------";
        monFlux.open("D:/H_01.txt",ios::app|ios::out);
      if(monFlux)
       {
           monFlux <<_ele._n<<endl;
            monFlux <<_ele._nom<<endl;
             monFlux <<_ele._prenom<<endl;
              monFlux <<s<<endl;
               monFlux.close();
            }
          else
         {
            cout << "ERREUR: Impossible d'ouvrir le fichier." << endl;
        }

}
//*************************************************************
void li_f()
{
       per _g;
        string s;
         cout<<""<<endl;
          fstream monFlux("D:/H_01.txt",ios::in);
        if(monFlux)
        {
            monFlux.seekg(0);
             monFlux >>_g._n>>_g._nom>>_g._prenom>>s;
              while(!monFlux.eof())
              {
              cout<<" * N      : "<<_g._n<<endl;
             cout<<" * Nom    : "<<_g._nom<<endl;
            cout<<" * Prenom : "<<_g._prenom<<endl;
        cout<<"   ----------------------------"<<endl;
       monFlux >>_g._n>>_g._nom>>_g._prenom>>s;
          }
     monFlux.close();
        }
   else
    {
     cout << "ERREUR: Impossible d'ouvrir le fichier en lecture." <<endl;
       }
}
///////////////////////////////////////////////////////////////////////////////////////////
```

```
int main(array<System::String ^> ^args)
{
       per p;
         int a,b=0;
           while(b==0)
             {
                 home();
                   cout<<" * Entrez le choi : ";
                     cin >> a;
       switch(a)
        {
      //------------------------------------------------------
      case 1:
        system("cls");
         try
          {
              cout<<" "<<endl;
               cout<<" * N       : ";
et1 : ;
               try
                {
                   cin >>p._n;
                    }
                     catch(...)
                      {
                      goto et1 ;
                         }
                         cout<<" * Nom     : ";
                        cout >>p._nom;
                      cout<<" * Prenom : ";
                   cin >>p._prenom;
                 ec_f(p);
                }
             catch(...)
            {
           cout << "   ERREUR : " <<endl;
        }
       break;
       //------------------------------------------------------
       case 2:
           system("cls");
                  li_f();
        break;
       //------------------------------------------------------
       case 3:
           system("cls");
             b=1;
                cout<<" "<<endl;
                  cout<<"        * Pour exit clik sur Entr :           "<<endl;
                    cout<<" "<<endl;
       break;
       //------------------------------------------------------
       default:
           cout<<" "<<endl;
             cout<<"          * CHOIX INEXISTANT :"<<endl;
               cout<<" "<<endl;
        break;
       }
       }
       Console::ReadKey();
     return 0;
}
```

3- Standard Template Library (STL):

الكثير من لغات البرمجة أصبحت توفر ضمن مترجماتها هده المكتبة (مكتبة القوالب القياسية) وهي توفر خدمات متميزة ورائعة في نفس الوقت . ألان وبعد العناء من التعامل مع المصفوفات الديناميكية والقوائم المرتبطة والمؤشرات . نجد ضمن هده المكتبة ما يجعلنا نقوم بعمل برامج كثيرة بكل سهولة . من خلال هدا الدرس سنتعرف علي عنصرين هامين من عناصر المكتبة وبعد دراسة البرمجة الكائنية التوجه سنعود لدراسة هده المكتبة بالتفصيل إنشاء الله .

➢ Victor :

هدا النوع يشبه المصفوفات الديناميكية إلي انه أفضل من حيث السهولة والأمن وسلامة التنفيذ . وهو يعمل من طرف واحد وهو الطرف الخلفي أي أن الإضافة والحدف تكون من الطرف الخلفي .

للتعامل مع هدا النوع من القوالب يجب تضمين المكتبة

```
#include <vector>
```

➢ La déclaration :

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <vector>

using namespace System;
using namespace std;
int main(array<System::String ^> ^args)
{
        vector <string> tab;

      Console::ReadKey();
      return 0;
}
```

صرحنا بمتغير من النوع **vector** باسم**tab** ثم بين العلامتين < > في هدا المثال اخترنا النوع **string** ادا اخترنا **int** نكتب بين العلامتين < **int** > ويمكن أن يكون من النوع **stract** .

يمكن التعامل معه كمصفوفة.

```
cout<<tab[i]<<endl;
```

```
#include "stdafx.h"
#include <iostream>
#include <vector>
#include <string>

using namespace System;
using namespace std;

int main(array<System::String ^> ^args)
{
        vector <string> tab;
        string s;
        for(int i=0;i<3;i++)
        {
                cin>>s;
                tab.push_back(s);
        }
        //---aff
        cout<<"------------------"<<endl;
        for(int i=0;i<3;i++)
        {
                cout<<tab[i]<<endl;
        }


    Console::ReadKey();
    return 0;
}
```

`tab.push_back(obj);`	إضافة عنصر من الطرف الخلفي .
`tab.pop_back();`	حد ف أخر عنصر من الطرف الخلفي.
`tab.begin();`	تعيد هده الدالة العنصر الأول.
`tab.end();`	تعيد هده الدالة العنصر الأخير.
`tab.erase(i);`	حذف العنصر دو الرقم i .
`tab.erase(i,j);`	حذف العناصر من i إلي j .
`tab.size();`	تعيد هده الدالة عدد العناصر الموجودة .
`tab.clear();`	تحذف هده الدالة جميع العناصر الموجودة .
`obj = tab[i];`	وضع العنصر الموجود في الرقم i في المتغير obj .

`int i,j;` نوع المتغيرين i و j

obj نوع المتغير الدي نضعه في vector

➢ List :

هدا النوع يشبه القوائم المترابطة (les lists) كما يوفر لنا الكثير من الدوال للتعامل معه . اغلب الدوال كثيرة الاستعمال ذكرت عند دراستنا لي العنصر vector بنفس الاسم وطريقة إرسال المتغيرات . إلي أن هدا النوع لديه إمكانية إضافة وحذف عنصر من الطرف الأمامي . نتعرف بعض الدوال المميزة له لاحقا .

للتعامل مع هدا النوع يجب تضمين المكتبة

```
#include <list>
```

```
#include "stdafx.h"
#include <iostream>
#include <string>
#include <list>

using namespace System;
using namespace std;

typedef list <string> var;
int main(array<System::String ^> ^args)
{
string s;
list <string> lt;
for(int i=0;i<3;i++)
{
  cin>>s;
  lt.push_back(s);
  cout<<endl;
}
cout<<"------------------------"<<endl;
for(var::const_iterator j = lt.begin();
j != lt.end();j++)
       {
             cout<<*j<<endl;
       }
    Console::ReadKey();
    return 0;
}
```

المتغير j عبارة عن مؤشر يشير إلي أول عنصر في القائمة في البداية ثم يتزايد بقدر حجم نوع المتغير الذي وضعناه في القائمة . ولإظهار جميع عناصر القائمة نحتاج إلي حلقة تكرارية لإظهار جميع القيم التي يشير إليها المؤشر j . عن طريق عرض (*j)

الأشياء الأخرى الغير مفهومة سنعود إليها بعد دراسة oop

lt.push_front(obj);	إضافة عنصر إلي أول القائمة .
lt.pop_front();	حذف أول عنصر في القائمة .

4- La gestion des exceptions :

```
#include "stdafx.h"
#include <iostream>

using namespace System;
using namespace std;

int main(array<System::String ^> ^args)
{
    int n;
    cin.exceptions(cin.failbit); //1
    cout << " * Entrez un nombre entier : ";
    try
   {
        cin >> n;
        cout <<" * Le nombre = " << n <<endl;
    }
    catch(...)
    {
             cout<<" * Erreur ?!!!!."<<endl;
    }
    cout<<" * fin ."<<endl;
    Console::ReadKey();
    return 0;
}
```

التعامل مع الأخطاء مهم جدا ولدلك سنفصله في وقت لا حق وهده لمحة بسيطة عن معالجة الأخطاء في C++ حيث أولا يجب أن نهيئ الكائن cin لاستقبال الأخطاء كما في العبارة 1 . ثم نكتب الكود الذي احتمال أن تقع فيه أخطاء(مثل أن يقوم المستخدم بإدخال سلسلة حرفية بدل عدد صحيح من المعلوم أن يتوقف البرنامج لكن مع هده الميزة يمكن تجاوز الأمر) داخل كتلة try. كما في المثال قراءة عدد صحيح وضعنها داخل كتلة try . في حالة وقوع خطاء ينتقل التنفيذ إلي كتلة catch وعرض رسالة خطأ مثلا . أما في حالة التنفيذ سليم يواصل البرنامج عمله بشكل عادي ولا ينفذ الكود داخل catch.